

Projet de réseaux et supervision avancés :

Réalisation d'un anémo giro sur NMEA2000 avec un ESP 32

I/ Prise en main.....	2
Déclarations et variables :.....	2
Fonction setup() :.....	2
anemoInterrupt() :.....	2
ReadWindAngle() :.....	2
ReadWindSpeed() :.....	2
SendN2kWind() :.....	3
II/ Recherche des ou du PGN concernés.....	4
III/ Génération de trames factices de vent pour l'ESP 32.....	5
IV/ Modification du Software.....	6
Calcul de la vitesse du vent.....	6
Calcul de la direction du vent.....	6
Visualisation des PGN et des valeurs sur le logiciel N2KAnalyzer.....	7
V/ Validation.....	8
Annexe : code complet.....	9

I/ Prise en main

Le programme est conçu pour lire des données provenant d'un anémomètre (mesure de la vitesse et de la direction du vent) et envoyer ces données via le protocole NMEA 2000. Il utilise des interruptions externes pour mesurer la période de rotation de l'anémomètre et pour calculer la vitesse du vent.

Déclarations et variables :

- *infini* : Un grand nombre utilisé pour indiquer une période très longue
- *windDir* : Broche pour mesurer la direction du vent
- *anemo* : Broche pour l'anémomètre
- *period, tWind, previousTWind* : Variables utilisées pour mesurer la période entre les impulsions de l'anémomètre (temps entre les rotations)
- *windDirection* : Direction du vent
- *windSpeed* : Vitesse du vent

Fonction setup() :

- Configuration des broches : Le pin pour la direction du vent est défini en entrée, et le pin de l'anémomètre est configuré sans résistance de pull-up (puisque le jumper est soudé).
- AttachInterrupt() : Pour mesurer les impulsions de l'anémomètre (sur déclenchement de la ligne RISING).
- NMEA2000 : Des informations sur le produit et l'appareil sont envoyées via le protocole NMEA 2000 pour identifier le capteur, la version du logiciel et du matériel, et la fonction ainsi que ses caractéristiques. Cela inclut l'ID du capteur.

anemolInterrupt() :

- Est appelée à chaque impulsion du capteur anémomètre (lors d'un changement d'état de la broche). Elle calcule la période entre deux impulsions successives en soustrayant le temps actuel (obtient via millis()) du dernier temps enregistré (previousTWind).

ReadWindAngle() :

- Lit la direction du vent en utilisant une lecture analogique sur le capteur connecté à *windDir*. La valeur lue est ensuite convertie en un angle entre 0 et 360 degrés.

ReadWindSpeed() :

- Calcule la vitesse du vent à partir de la période mesurée par l'anémomètre en utilisant la formule empirique (formule modifiée dans la suite du TP) " $windSpeed = 2132.5 * period^{-0.88}$ ", où la vitesse du vent est inversement proportionnelle à la période de rotation de l'anémomètre.

SendN2kWind() :

- Envoie les données de vitesse et de direction du vent à intervalles réguliers (1 seconde puis 100ms au cours des modifications). Utilise la fonction SetN2kWindSpeed() pour créer un message NMEA2000 et le transmettre sur le bus.

II/ Recherche des ou du PGN concernés

Le PGN qui correspond aux valeurs recherchées est le “PGN 130306” nommé “Wind data”

Wind Data		PGN: 130306			
		hex: 1FD02			
Direction and speed of Wind. True wind can be referenced to the vessel or to the ground. The Apparent Wind is what is felt standing on the (moving) ship, i.e., the wind measured by the typical mast head instruments.					
The boat referenced true wind is given by the vector sum of Apparent wind and vessel's heading and speed though the water. The ground referenced true wind is given by the vector sum of Apparent wind and vessel's heading and speed over ground.					
Single Frame:	Yes	Priority Default:	2		
Destination:	Global	Query Support:	Optional		
Field #	Field Name	Original Reference ID # 81			
1	Sequence ID	Byte Field Size:	1		
		Request Parameter	Optional		
		Bit Field Size:	Optional		
DD056	Sequence ID	An upward counting number used to tie related information together between different PGNs . For example, the SID would be used to tie together the COG, SOG and RAIM values to a given position. 255=no valid position fix to tie it to. Range 0 to 252 for valid position fixes.			
DF53	Integer, 8 bit unsigned	uint8	Range: 0 to 252 Resolution: 1 bit Unit-less number		
2	Wind Speed	Byte Field Size:	2		
		Request Parameter	Optional		
		Bit Field Size:	Optional		
DD044	Generic Speed				
DF35	Speed	uint16	Range: 0 to 655.32 m/s Resolution: 1x10E-2 m/s 1 Knot = 0.5144 m/s		
3	Wind Direction	Byte Field Size:	2		
		Request Parameter	Optional		
		Bit Field Size:	Optional		
DD045	Wind Direction				
DF02	Angle	uint16	Range: 0 to 2Pi rad Resolution: 1x10E-4 rad Resolution ~0.0057deg, 1 deg = .01745 rad		
4	Wind Reference	Byte Field Size:	3		
		Request Parameter	Optional		
		Bit Field Size:	Optional		
DD205	Wind Reference	0x00 = Theoretical Wind (ground referenced, referenced to True North; calculated using COG/SOG) 0x01 = Theoretical Wind (ground referenced, referenced to Magnetic North; calculated using COG/SOG) 0x02 = Apparent Wind (relative to the vessel centerline) 0x03 = Theoretical (Calculated to Centerline of the vessel, referenced to ground; calculated using COG/SOG) 0x04 = Theoretical (Calculated to Centerline of the vessel, referenced to water; calculated using Heading-Speed through Water) 0x05 = Reserved 0x06 = Error			

III/ Génération de trames factices de vent pour l'ESP 32

Ce projet consiste en la modification d'un programme nommé "WindMonitorReal", initialement codé pour AtMega. Les Modifications de broches et de bibliothèques sont réalisées pour assurer la compatibilité avec l'esp :

WindMonitorRealRefait.ino

```

1  #define ESP32_CAN_TX_PIN GPIO_NUM_5
2  #define ESP32_CAN_RX_PIN GPIO_NUM_4
3
4  #include <Seasmart.h>
5  #include <N2kMsg.h>
6  #include <NMEA2000_CompilerDefns.h>
7  #include <ActisenseReader.h>
8  #include <N2kGroupFunctionDefaultHandlers.h>
9  #include <N2kTimer.h>
10 #include <N2kMaretron.h>
11 #include <N2kStream.h>
12 #include <N2kMessages.h>
13 #include <N2kDef.h>
14 #include <N2kGroupFunction.h>
15 #include <N2kCANMsg.h>
16 #include <NMEA2000.h>
17 #include <NMEA2000_CAN.h>
18 #include <N2kDeviceList.h>
19 #include <N2kMessagesEnumToStr.h>
20 #include <RingBuffer.h>
21 #include <NMEA2000StdTypes.h>
22 #include <N2kTypes.h>
23
24 #include <ESP32_CAN_def.h>
25 #include <NMEA2000_esp32.h>
26
27 // Demo: NMEA2000 library. Send main wind data to the bus.
28
29 #include <Arduino.h>
30

// Pins remapped
#define P_MOSI          23
#define P_MISO          19

//----- GIRO ANEMO -----
#define infini 1000000L
#define windDir 34
#define anemo 15

int etalonnage;
unsigned long period=infini, t0=0, t1=0, tWind=0, previousTWind=0;
int convert, windDirection, te = 1000, bouc=0;
float windSpeed;

void loop() {
    t1= millis();
    if(( t1 - tWind) > 5000)
        period = infini;
    SendN2kWind();
    NMEA2000.ParseMessages();
}

void setup() {
    pinMode(windDir, INPUT);
    pinMode(anemo, INPUT);

    sei();
    delay(200);
    etalonnage = pow(2,12);
}

```

Le temps limite dans la loop sert à tirer vers zéro pour ne pas conserver une valeur de vitesse alors que l'anémomètre est imobile.

IV/ Modification du Software

Calcul de la vitesse du vent

```
double ReadWindAngle() {
    convert = analogRead(windDir);
    windDirection = constrain(((convert * 360.0) / etalonnage), 0, 360); //1.0 coeff pour arriver a 360
    return DegToRad(windDirection);
}
```

Calcul de la direction du vent

```
double ReadWindSpeed() {
    windSpeed = (2.25 / (period * 0.001)) / 1.151;
    return windSpeed;
}
```

Il est ainsi possible de voir apparaître l'anémomètre sur le réseau :

Maretron N2KAnalyzer, Version 2.4.22.3 - VesselName												
File Setup Analyze Update Configure Web Help												
Expand	Node Address	Manufacturer	Mfg Model ID	Mfg Serial Number	Source	Unique Instance	Label	Current Software	Available Software	Installation Description #1	Installation Description #2	Bandwidth 2.8%
55	Maretron	USB100	1171267		0		2.1.1.2	2.1.1.2	Passerelle			0.0%
05	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-				0.0%
01	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-				1.1%
02	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-				0.0%
03	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-				0.0%
04	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-				0.0%
00	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-				0.0%
2B	#427	R300	3820181		0		060200.0...	-				0.0%
53	Maretron	USB100	1171264	0			2.1.1.2	2.1.1.2				0.0%
24	Victron	SmartSolar M...	0291997 HQ2...		0		3.14	-				0.8%
70	Maretron	TLM100	1508163	Fuel	0	Reservoir babord	2.0.2.3	2.0.2.3				0.0%
60	Maretron	SSC300	1781549		0	gyro-compas	1.0.6.1	1.0.6.1				0.4%
19	Maretron	DSM410	1881507		0	écran	1.8.3.2	1.8.1.2				0.0%
23	Airmar	DST XXX	OEM-SerialC...		0		3.005.3.0...	-				0.5%
0A	Maretron	DSM410	1881509		0	écran	1.8.3.2	1.8.1.2				0.0%
17	#2046	wind monito...	00000002		0		1.0.0 (20...	-				0.1%

(ligne surlignée en bleu)

Visualisation des PGN et des valeurs sur le logiciel N2KAnalyzer

The screenshot shows the Maretron N2KAnalyzer software interface. At the top, there is a menu bar with File, Setup, Analyze, Update, Configure, Web, and Help. Below the menu is a toolbar with various icons. The main window contains a table of network nodes:

Expand	Node Address	Manufacturer	Mfg Model ID	Mfg Serial Number	Source	Unique Instance	Label	Current Software	Available Software	Installation Description #1	Installation Description #2	Bandwidth
	55	Maretron	USB100	1171267		0		2.1.1.2	2.1.1.2	Passerelle		2.8%
	05	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-			0.0%
	01	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-			1.1%
	02	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-			0.0%
	03	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-			0.0%
	04	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-			0.0%
	00	#595	MiniPlex-3Wi...	3B001166		0		3.4.0	-			0.0%
	2B	#427	R300	3B20181		0		060200.0...	-			0.0%
	53	Maretron	USB100	1171264		0		2.1.1.2	2.1.1.2			0.0%
	24	Victron	SmartSolar M...	0291997 HQ2...		0		3.14	-			0.8%
	70	Maretron	TLM100	1508163	Fuel	0	Reservoir babord	2.0.2.3	2.0.2.3			0.0%
	60	Maretron	SSC300	1781549		0	gyro-compas	1.0.6.1	1.0.6.1			0.4%
	19	Maretron	DSM410	1881507		0	écran	1.8.3.2	1.8.1.2			0.0%
	23	Airmar	DST XXX	OEM-SerialC...		0		3.005,3.0...	-			0.5%
	0A	Maretron	DSM410	1881509		0	écran	1.8.3.2	1.8.1.2			0.0%
	17	#2046	wind monitor...	00000002		0		1.0.0 (20...	-			0.1%

A modal window titled "TX wind monitor antoine leo (0x17) 00000002 - Transmitted PGNs" is open, showing a list of transmitted PGNs:

Time	PGN	Description
2018.93	60928	ISO Address Claim
1846.88	126464	PGN List - Transmit PGNs group function
1846.70	126464	PGN List - Received PGNs group function
1374.67	126593	Heartbeat
1846.47	126596	Product Information
1846.57	126598	Configuration Information
2024.67	130306	Wind Data

Details for the last transmitted PGN (2024.67, 130306) are shown:

- Destination: Global
- Sequence ID: 1
- Wind Speed: 0.01944 kt
- Wind Direction: 0.00000 °M
- Wind Reference: Apparent (relative to the vessel centerline)
- NMEA Reserved: 2097120

VI/ Validation

Voici la vitesse du vent ainsi que la direction du vent sur l'écran DSM410 Maretron



l'écran Maretron est également émulé sur le logiciel N2KAnalyzer pour afficher les valeurs “Wind Apparent Direction” et “Wind Apparent Speed”



Annexe : code complet

```
#define ESP32_CAN_TX_PIN GPIO_NUM_5
#define ESP32_CAN_RX_PIN GPIO_NUM_4

#include <Seasmart.h>
#include <N2kMsg.h>
#include <NMEA2000_CompilerDefns.h>
#include <ActisenseReader.h>
#include <N2kGroupFunctionDefaultHandlers.h>
#include <N2kTimer.h>
#include <N2kMaretron.h>
#include <N2kStream.h>
#include <N2kMessages.h>
#include <N2kDef.h>
#include <N2kGroupFunction.h>
#include <N2kCANMsg.h>
#include <NMEA2000.h>
#include <NMEA2000_CAN.h>
#include <N2kDeviceList.h>
#include <N2kMessagesEnumToStr.h>
#include <RingBuffer.h>
#include <NMEA2000StdTypes.h>
#include <N2kTypes.h>

#include <ESP32_CAN_def.h>
#include <NMEA2000_esp32.h>

// Demo: NMEA2000 library. Send main wind data to the bus.

#include <Arduino.h>

// List here messages your device will transmit.
const unsigned long TransmitMessages[] PROGMEM={130306L,0};

// Pins remapped
#define P_MOSI 23
#define P_MISO 19

//----- GIRO ANEMO -----
#define infini 1000000L
#define windDir 34
#define anemo 15
```

```
int etalonnage;
unsigned long period=infini, t0=0, t1=0, tWind=0, previousTWind=0;
int convert, windDirection, te = 1000, bouc=0;
float windSpeed;

void anemoInterrupt(void);

void setup() {
    pinMode(windDir, INPUT);
    pinMode(anemo, INPUT);

    sei();
    delay(200);
    etalonnage = pow(2,12);

    // "External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt
    5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2)"
    attachInterrupt(digitalPinToInterruption(anemo), anemoInterrupt,
RISING); //parce que ca marchait pas en falling

    // Set Product information
    NMEA2000.SetProductInformation("00000002", // Manufacturer's Model serial
code
                                    100, // Manufacturer's product code
                                    "wind monitor antoine leo", //
Manufacturer's Model ID
                                    "1.0.0 (2020-12-10)", // Manufacturer's
Software version code
                                    "1.0.0 (2020-12-10)" // Manufacturer's Model
version
                                    );
    // Set device information
    NMEA2000.SetDeviceInformation(1, // Unique number. Use e.g. Serial number.
                                    130, // Device function=Atmospheric. See
codes on
http://www.nmea.org/Assets/20120726%20nmea%202000%20class%20&%20function%20co
des%20v%202.00.pdf
                                    85, // Device class=External Environment. See
codes on
http://www.nmea.org/Assets/20120726%20nmea%202000%20class%20&%20function%20co
des%20v%202.00.pdf
```

```
2046 // Just choose from code list on
http://www.nmea.org/Assets/20121020%20nmea%202000%20registration%20list.pdf
);

// Uncomment 2 rows below to see, what device will send to bus. Use e.g.
OpenSkipper or Actisense NMEA Reader
//Serial.begin(115200);
//NMEA2000.SetForwardStream(&Serial);
// If you want to use simple ascii monitor like Arduino Serial Monitor,
uncomment next line
//NMEA2000.SetForwardType(tNMEA2000::fwdt_Text); // Show in clear text.
Leave uncommented for default Actisense format.

// If you also want to see all traffic on the bus use N2km_ListenAndNode
instead of N2km_NodeOnly below
NMEA2000.SetMode(tNMEA2000::N2km_NodeOnly,23);
// NMEA2000.SetDebugMode(tNMEA2000::dm_Actisense); // Uncomment this, so
you can test code without CAN bus chips on Arduino Mega
NMEA2000.EnableForward(false);
NMEA2000.ExtendTransmitMessages(TransmitMessages);
NMEA2000.Open();
}

//ISR(INT5_vect)
void anemoInterrupt(void)
{
    tWind = millis();
    period = tWind - previousTWind;
    previousTWind = tWind;

}

void loop() {
    t1= millis();
    if(( t1 - tWind) > 5000)
        period = infini;
    SendN2kWind();
    NMEA2000.ParseMessages();
}

double ReadWindAngle() {
    convert = analogRead(windDir);
    windDirection = constrain(((convert * 360.0) / etalonnage), 0, 360); //1.0
    coeff pour arriver à 360
    return DegToRad(windDirection);
}
```

```
double ReadWindSpeed() {
    windSpeed = (2.25 / (period * 0.001)) / 1.151;
    return windSpeed;
}

#define WindUpdatePeriod 100

void SendN2kWind() {
    static unsigned long WindUpdated=millis();
    tN2kMsg N2kMsg;

    if ( WindUpdated+WindUpdatePeriod<millis() ) {
        SetN2kWindSpeed(N2kMsg, 1, ReadWindSpeed(), ReadWindAngle(),
        N2kWind_Apprent);
        WindUpdated=millis();
        NMEA2000.SendMsg(N2kMsg);
    }
}
```